Journal of Computer and Communications, 2020, 8, 107-119
https://www.scirp.org/journal/jcc
ISSN Online: 2327-5227
ISSN Print: 2327-5219

A Receiver-Initiated Approach with Fuzzy Logic Control in Load Balancing

Mingchang Huang

Department of Business Information System/Management Operations, University of North Carolina at Charlotte, Charlotte, NC, USA
Email: mhuang5@uncc.edu

How to cite this paper: Huang, M.C. (2020) A Receiver-Initiated Approach with Fuzzy Logic Control in Load Balancing. *Journal of Computer and Communications*, **8**, 107-119.
https://doi.org/10.4236/jcc.2020.85007

Received: March 31, 2020
Accepted: May 25, 2020
Published: May 28, 2020

Abstract

This paper proposes a new approach for load balancing by using receiver-initiated load transfer method. Usually data transfer for load balancing starts from a sender node. This is what called a sender-initiated method. In this paper, instead, a load balancing action starts from a receiver node; the receiver node initiates load balancing for asking a sender node for load transfer. Fuzzy logic control is applied in this approach to avoid using a fixed threshold value in load balancing in ad-hoc networks. Performance for the receiver-initiated approach is evaluated and compared with other load balancing approaches—BID protocol, fuzzy logic sender-initiated algorithm and non-fuzzy (threshold based) receiver-initiated algorithm. The results show that the receiver-initiated approach improves network performance by comparing with the other three.

Keywords

Load Balancing, Receiver-Initiated, Sender-Initiated, Fuzzy Logic Control

1. Introduction

Computer networks can provide computation parallelism. However, an imbalance in workloads among computers reduces the performance of these systems. To take advantage of the full capacity of these computer systems, load balancing and load sharing algorithms have been devised to improve the performance of these systems by the appropriate transfer of tasks among available computers.

There are several algorithms for load balancing and load sharing. Most of them use local information for transferring excessive loads from heavily loaded nodes to lightly loaded nodes. Moreover, fixed threshold levels are used in those works to decide whether a node is heavily or lightly loaded [1]. An alternative approach to improve the system performance is to use fuzzy logic control in a

distributed network system [2] [3].

In general, there are two different ways to balance loads among computers [4]-[13]—static and dynamic. 1) In the static method, threshold levels are fixed and are not changed according to the current status of the system. Therefore, nodes in the system do not exchange state information for choosing new threshold level(s). 2) In contrast in the dynamic method, threshold levels are changed according to the current status of the system and state information exchange is a necessary part of this approach. Dynamic methods generally react better to changes in the system state compared to the static methods and as a result have better performance.

Usually, there are two main methods for load balancing—sender-initiated and receiver-initiated methods. For sender-initiated methods, a heavily loaded computer initiates load balancing approaches for load transfer. The sender sends request to the network to ask nodes for load transfer [3] [14]. On the contrary, a lightly loaded computer initiates load balancing in receiver-initiated methods. I have showed in [3] that the sender-initiated method using fuzzy logic control method has a better performance than conventional methods, and further it is more stable than the algorithm using non-fuzzy logic control (*i.e.* algorithm using discrete threshold levels). In this paper, I use receiver-initiated method with fuzzy logic control for load balancing and compare the system performance with other algorithms.

Section 2 introduces the fuzzy logic control concept. Section 3 discusses the system model. Section 4 includes the details of the proposed protocol. Section 5 shows preliminary simulation results and Section 6 presents the conclusion.

2. Fuzzy Logic Control

In this section, I introduce the basic concept of fuzzy logic control proposed by Zadeh in 1965. For a good explanation, see [15]. Fuzzy logic is a powerful tool in representing linguistic information and is very useful to solve problems that do not have a precise solution and the conventional methods cannot solve them very well. For example, a computer load can be regarded as a linguistic variable and its value can be considered as light, moderate, and heavy. **Figure 1** shows an example, where if the load is between 0 and 30, or between 10 and 70, or more than 50, then the computer is considered as lightly, moderately, or heavily loaded respectively. In this figure, the horizontal axis represents a computer load and the vertical axis shows the membership function and the degree a computer load is lightly, moderately, or heavily loaded. For example, point A in this figure represents that the computer load is 100% lightly loaded, in contrast to the point B where the computer load is considered 80% lightly loaded and 40% moderately loaded.

In the conventional load balancing schemes with a single threshold level, a computer system load status is considered either lightly loaded or heavily loaded depending on whether its load is below or above a threshold level. Therefore, a

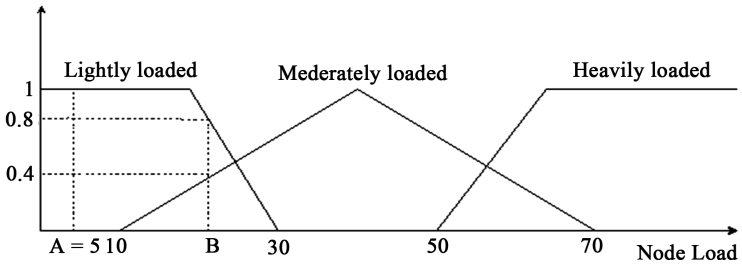

Figure 1. Fuzzy functions for the load of nodes.

computer system, which is lightly loaded, may suddenly become heavily loaded by receiving an extra load. In contrast, in a fuzzy logic control, a computer system load status gradually changes from the lightly loaded status to the heavily loaded status depending on its membership function that represents a more realistic change of load status than conventional methods. Hence there is a greater potential for system stability due to reducing unnecessary jobs movements and reducing thrashing probability.

Basically, a fuzzy logic control scheme includes inputs, an inference mechanism, and an output. After the input variables are determined, a rule based engine can be set in the form of "IF A and B THEN C", where A, B, and C are fuzzy sets. Figure 2 shows the fuzzy logic controller structure that consists offuzzifier, inference machine, defuzzifier, and rule base components.

Fuzzifier converts the input values to degrees of membership via the membership functions. Usually, the degrees are between 0 and 1. Inference machine implements rule references according to the fuzzy rule base in order to generate outputs. Inference rules use the MAX-MIN method to generate the output degrees. Defuzzifier then uses the output degrees to calculate the output crisp value.

3. System Model

I assume that underlying computer networks have a hierarchical structure and use the system model in [3]. For example, as shown in Figure 3, several nodes form a group and each group contains a node called designated representative (DR), which communicates with the DRs in other peer groups. Group 1contains three nodes—A.1.1, A.1.2, and A.1.3, where A.1.2 is the DR in this group. Group 2 contains four nodes – A.2.1, A.2.2, A.2.3, and A.2.4 and A.2.2 is the DR of the group. Group 3 contains four nodes—B.1, B.2, B.3 and B.4, where B.1 is the DR of the group. All these three groups are physical groups. Group 4 is a logical group at the next level of the hierarchy. Each node in a logical group is a logical node that represents the whole lower group for the next level in the hierarchy. For example, logical node A.1 represents Group 1 and logical node A.2 represents Group 2.

This hierarchical structure is similar to the computer networks structure where each of the physical groups 1, 2, and 3 may represent a Local Area Network (LAN), and each of the next level groups, like Group 4, may represent a

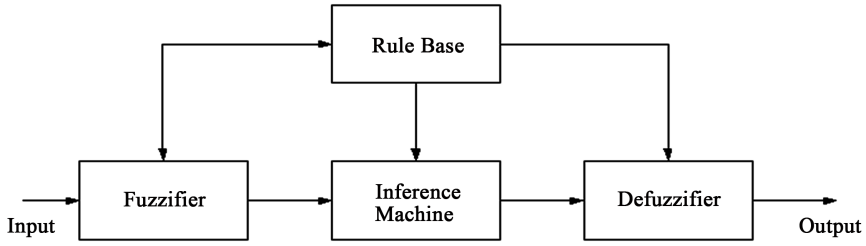

Figure 2. Fuzzy logic controller architecture.

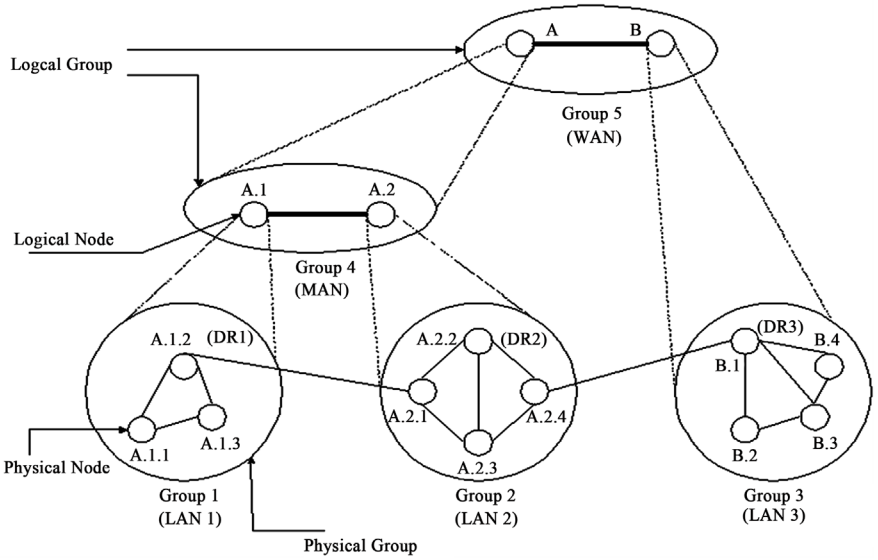

Figure 3. System configuration for load balancing.

Metropolitan Area Network (MAN), and each of the higher groups, like Group 5, may represent a Wide Area Network (WAN). This structure is similar to Internet and Asynchronous Transfer Mode (ATM) network structures. In this model, it is assumed that logical nodes at upper levels are represented by the physical nodes at the lowest levels. For example, the physical nodes A.1.2 and A.2.2 represent logical nodes A.1 and A.2 respectively.

4. Receiver-Initiated Fuzzy-Logic Load Balancing Approach

The approach is composed of the following main steps.

4.1. Group Forming

In order to reduce the communication overhead, nodes at close distance may form a group. For instance, all or some of the nodes in the same Local Area Network (LAN) may form a group. In the system model, nodes may join or leave their groups dynamically. A node may join a group by sending a message to its group's DR. For a node to leave a group, the node has to send its entire load to some other nodes. This step is implemented as follows:

- The leaving node sends a "LEAVE" message to its DR about its leaving decision.

- Then the DR finds suitable nodes for load transfer.
- The leaving node will transfer its loads in its queue without accepting loads at the same time. If it's loads are not completely transferred after a certain time period, then the remaining jobs in the queue are dropped. The time period is variable and, for instance, set to at least twice the maximum allowance of waiting time for making a load transfer request.

4.2. Fuzzy Logic Load Balancing Algorithm

The following are main the steps for the algorithm.

- *Fuzzification*

Every node should evaluate its own load status as new jobs came in. First, the workload of a node is determined. Then it is used as an input to its membership functions for finding its degree according to a table, which is explained later.

- *Applying the inference rules*

A lightly loaded node may send a request for finding a heavily loaded node for the load transfer using the fuzzy logic inference rules.

- *Defuzzification*

I use the centroid method [15] to determine the output crisp value (the load transfer probability). Once the workloads for the sender and receiver are known and the inference rule is applied, then the output crisp value is found from the output membership functions.

- *Find the suitable node for the load transfer*

The easiest way for a heavily node to find a suitable node for load transfer is to use the first request node it receives. There might be several nodes that are lightly loaded to ask for load transfer and the requesting node may choose the least lightly loaded node. One method is to choose the node that has the highest crisp value after the defuzzification process.

- *Transfer the loads from the sender to the receiver*

It is possible that some of the senders may request the same receiver at the same time. The system performance may degrade if the receiver accepts all the requests. To prevent this to happen, the receiver may choose only the first request and reject the others.

4.3. Fuzzy Logic Control

Two input variables are used – the current load of the sender and the current load of the receiver. The output is the probability for the load transfer. The bigger the output is, the higher the probability to transfer the loads of the node. Five fuzzy subsets for the input variables are defined—VL, L, M, H, and VH, which represents very lightly loaded, lightly loaded, moderately loaded, heavily loaded, and very heavily loaded states respectively. Similarly, five subsets are defined for the output variable—VL, L, M, H, and VH, which similarly represent the very low, low, medium, high, and very high probabilities for load transfer between the two nodes. The membership functions for input and output variables are shown

in Figure 4 and Figure 5 separately, and the rules for the rule base are shown in Table 1.

For example, if a sender is lightly loaded, then the probability that it probes and balances its load with a receiver, which is moderately loaded, is very low according to the reference rules in the Table 1. But when the probing sender is heavily loaded and the probed receiver is very lightly loaded, the probability that they balance their loads is high. In this case, if the output crisp value (output probability) is 0.9, there is a 90% chance (probability) that the heavily loaded sender balances its loads with the lightly loaded receiver.

4.4. Receiver-Initiated Load Balancing Protocol

An end node is a leaf node in the hierarchical structure of the network. It is assumed that the system is homogeneous, in the sense that every node has the

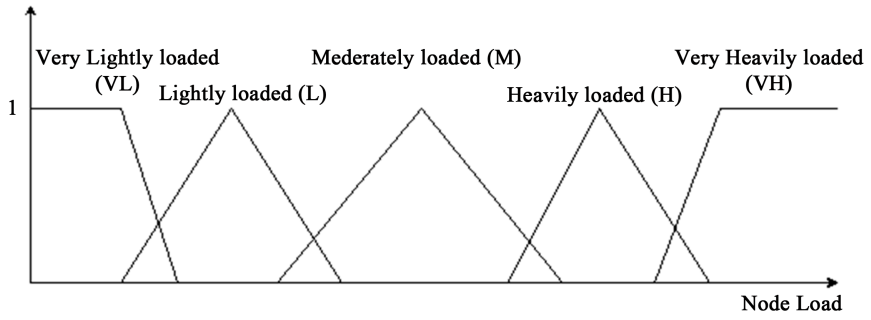

Figure 4. Membership functions for input variables.

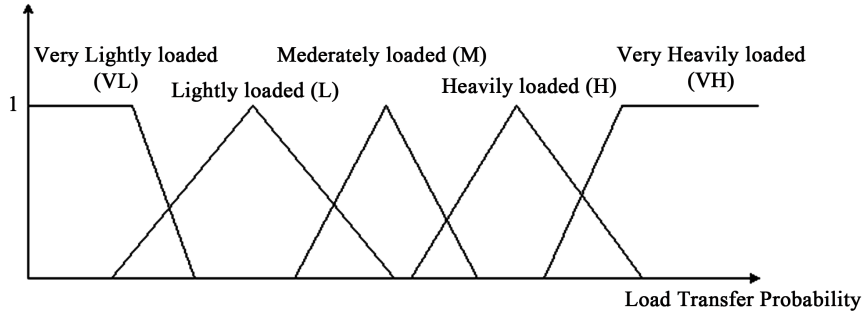

Figure 5. Membership functions for output variables.

Table 1. Inference rules for fuzzy logic control in load balancing.

Receiver Load	*Sender load*				
	VL	L	M	H	VH
VL	VL	L	M	H	VH
L	VL	VL	L	M	H
M	VL	VL	VL	L	M
H	VL	VL	VL	VL	L
VH	VL	VL	VL	VL	VL

same membership function and each node can use that membership function. Contrary to the sender-initiated protocol, in receiver-initiated protocol, an end node sends a load transfer request to other nodes when it is lightly loaded. A node checks its load status using fuzzy logic control first. If it is lightly loaded, then it uses multicast method to send a request to other nodes in its own group asking for load transfer. If there is no heavily loaded node available in its own group, then the node sends a request to the DR of its group for finding a lightly loaded node in other groups. Next, the DR of its group randomly chooses a DR in the same logical group and sends a load transfer request to that DR. After receiving that request, that DR multicasts a load transfer request to the nodes in its group.

The algorithm for the DR is similar to the sender-initiated protocol except that the DR will continue the search until either it successfully finds a heavily loaded node or exhaustively searches the entire network hierarchical tree structure.

4.4.1. Algorithm for a DR Node Using Receiver-Initiated Protocol

When a DR node receives a "LOAD-TRANSFER" message, it will perform the following:

1) When a DR node receives the "LOAD-TRANSFER" message from an End-Node within its own group, then the DR node sends a "LOAD-TRANSFER" message to another DR node in its logical group.

2) When a DR node receives a "LOAD-TRANSFER" from another DR, then it multicasts a "LOAD-TRANSFER" message to the End-Nodes within its group.

3) The DR node, which has sent a "LOAD-TRANSFER" message, waits for a time out equal to the maximum round trip delay in the network.

3a) If the DR node receives a "FOUND" message, then it terminates the search and relays the "FOUND" message to the last DR that sent a "LOAD-TRANSFER" message.

3b) If the DR does not receive a "FOUND" message within a time out equal to the maximum round trip delay in the network, then it should poll another DR within its logical group if one exists. Then it sends a "LOAD-TRANSFER" message to that DR it chooses, or else it sends a "LOAD-TRANSFER" message to its own DR in the next upper level in the network hierarchical tree structure, which will follow similar steps recursively.

3c) The DR node should continue the search until either it successfully finds a heavily loaded node or exhaustively searches the entire network hierarchical tree structure.

4.4.2. Algorithm for an End Node Using Receiver-Initiated Protocol

The algorithm for an end node is explained as below.

1) Multicast a "LOAD-TRANSFER" message to all the nodes within its own group.

2) Wait to receive replies from the nodes within the group which are heavily or very heavily loaded defined according to the fuzzy logic control scheme. The

wait time could be equal to the maximum round trip delay in the network.

3) A heavily loaded node may receive different load transfer requests from different lightly loaded nodes. Then this heavily loaded node chooses the least lightly node (defined according to the fuzzy logic control) for load transfer and sends an acknowledgement to that lightly loaded node.

4) If a lightly loaded node is chosen by a heavily loaded node, then it runs the fuzzy logic rule based control to get the output crisp value P—the probability for the load transfer. Next, this lightly loaded node gets M loads from the heavily loaded node where $M = P * (LS - LR)/2$, LS is the load of the sender and LR is the load of the receiver.

5) Otherwise, the lightly loaded node sends a "LOAD-TRANSFER" message to the DR of its group. Then the DR sends this message out to other DRs within its own group for finding some other heavily loaded end nodes in other groups (as was explained in the DR protocol) for load transfer.

5a) If a heavily loaded node is found in other group, then run the fuzzy logic rule based control to get the output crisp value P and transfer M loads to the lightly loaded node, where $M = P * (LS - LR)/2$. And also send a "FOUND" message to the corresponding DR.

Example: Assume that all the nodes in Group 1 (LAN 1) in **Figure 3** are lightly loaded. Thus a lightly loaded node A.1.1 has to find another node in other groups for load transfer. First node A.1.1 sends a request to its DR (node A.1.2) for load transfer. Logical node A.1 in Group 4 represents the nodes in Group 1. Logical node A.1 (acting as the DR in Group 1) sends a request to the logical node A.2 in Group 4 for finding a heavily loaded node in Group 2. Logical node A.2 represents the nodes in Group 2 (LAN 2) and acts as the DR of Group 2. Therefore, A.2.2 (the DR of Group 2) uses multicast to send requests to all the nodes within Group 2. If there is a heavily loaded node in Group 2, then it will reply directly to the original node A.1.1 that sent the load transfer request. At this time, that heavily loaded node starts transferring loads directly to the original node A.1.1 and sends a "FOUND" message to its DR to stop the search. If that heavily loaded node receives more than one request from other lightly loaded nodes during a period, then it selects the least lightly loaded node and sends an acknowledgement to that lightly loaded node for load transfer.

5. Preliminary Simulation Results and Analysis

5.1. Simulation Environment

Network simulator NS-2 [16] is used for the simulation study. **Figure 6** shows the system used for the simulation. There are three groups in the system with 3 nodes in each group. Each group represents a local area network. Transport layer protocol UDP (User Datagram Protocol) [17] is used for this study. The communication speed for each link is considered to be 5Mb/sec. Jobs arriving to nodes are assumed to have exponentially distributed service and inter-arrival time. It is assumed that the mean job arrival rate for each node is λ, the service

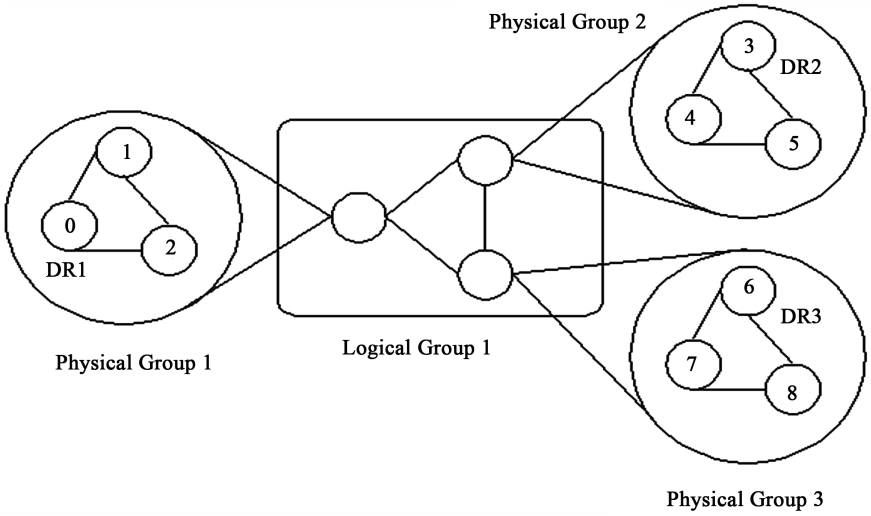

Figure 6. System model for simulation.

time for each node is S, and the utilization for that node is U = λS. I assume that S is equal to 50 time units and the utilization for all the nodes in the system is less than 1 except for Node 1 in Physical Group 1 and Node 4 in Physical Group 2, which generate jobs more than what their queues can store. This forces these nodes to transfer some of their loads to other nodes when their loads are above the heavily loaded region according to the fuzzy logic control. It is also assumed that arriving jobs are dropped when the node's queue is full.

5.2. Performance Comparison

In this study, I compare the performances of the following three algorithms—BID [18], Non-Fuzzy Logic Receiver-initiated and Fuzzy Logic Receiver-initiated algorithms.

5.2.1. BID Algorithm

A node multicasts a request to the other nodes in the same group when it is lightly loaded. Each of the other nodes sends an acknowledgement back to that node when each receives the request and is heavily loaded. If none of the nodes in the group are available for load transfer, then the requesting node waits for some period of time and then re-sends the request again instead of sending its request to the nodes in other groups.

5.2.2. Non-Fuzzy Logic Receiver-Initiated Algorithm

The Non-Fuzzy Logic Receiver-initiated algorithm uses the same system structure and the transfer protocol as I described in Section 3 and Section 4. However, it does not apply fuzzy logic control; instead, it uses fixed threshold levels to decide whether a node is heavily or lightly loaded. Unlike the BID algorithm that sends requests only to the nodes within a group, the non-fuzzy logic receiver-initiated algorithm may send requests to other nodes in different groups to ask for load transfer.

5.3. Performance Comparison of Receiver-Initiated Protocol

The following figures compare performances of these three algorithms. Figure 7 and Figure 8 show the comparisons of execution times and drop rates for these load balancing algorithms running 5000 jobs executed. From Figure 7, I can see that the system performance using fuzzy receiver-initiated algorithm is the best among these three algorithms. BID algorithm is the least efficient one among them according to the data collected. The Fuzzy Logic Receiver-initiated algorithm outperforms Non-Fuzzy Logic Receiver-initiated algorithm then system utilization is less than 0.7, and they almost have the same system performance when utilization is higher than 0.7.

With respect to the drop rate comparison among these algorithms, the Fuzzy Logic Receiver-initiated algorithm still outperforms the two other algorithms. Figure 8 shows that the drop rate for Fuzzy Logic Receiver-initiated algorithm is lower than that using Non-Fuzzy Logic Receiver-initiated algorithm, especially when the utilization is low. This indicates that the system is more stable using Fuzzy Logic Receiver-initiated algorithms than using Non-Fuzzy Logic Receiver-initiated algorithms.

5.4. Performance Comparison of Receiver-Initiated and Sender-Initiated Protocols

In this section, I compare the system performance under the Fuzzy Logic Sender-initiated and Receiver-initiated protocols. The results are shown in Figure 9 and Figure 10. It is shown in Figure 9 that the system performance using Fuzzy Logic Sender-initiated protocol is better than that using Fuzzy Logic Receiver-initiated protocol. Figure 10 shows the result of drop rate for both protocols. Similarly, the drop rate for Fuzzy Logic Sender-initiated protocol is a little bit lower than that using Fuzzy Logic Receiver-initiated protocol when utilization is higher than 0.3. This means that a system under the Fuzzy Logic Sender-initiated protocol is more stable than that under the receiver-initiated protocol when system has high utilization.

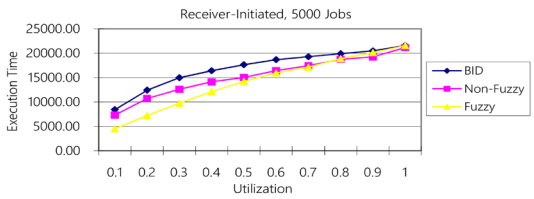

Figure 7. Performance comparison for receiver-initiated protocol.

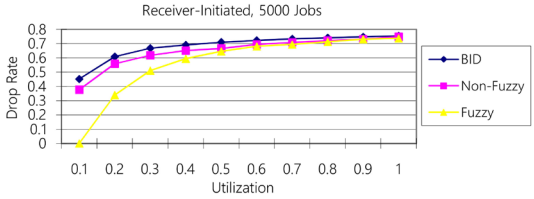

Figure 8. Drop Rate Comparison for Receiver-Initiated protocol.

The results in Figure 9 are slightly different from the results in [19] for the following two reasons. First, I use multicast in the protocols where a heavily loaded node (for Fuzzy Logic Sender-initiated protocol) or a lightly loaded node (for Fuzzy Logic Receiver-initiated protocol) only sends a request out to every node asking for load transfer, instead of sending requests to each nodes one by one as is done in conventional methods [19]. This reduces the probing time for a node to find other nodes for load exchanging for both sender-initiated and receiver-initiated methods; which means that there is no time difference for these two protocols to probe other nodes. Secondly, the number of ACKs transferred in the Fuzzy Logic Sender-initiated protocol is less than that in the Fuzzy Logic Receiver-initiated protocol. For example, only one ACK is needed for the Fuzzy Logic sender-initiated protocol, while it needs two ACKs for the Fuzzy Logic Receiver-initiated protocol. Figure 11 shows the steps and ACKs for both protocols.

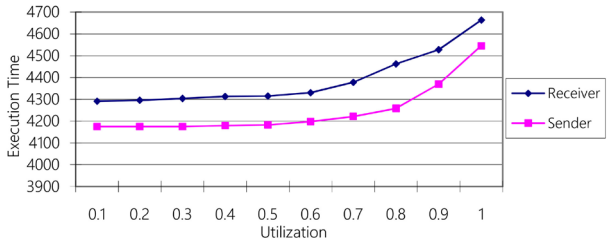

Figure 9. Performance comparison for sender-initiated protocol and receiver-initiated protocol.

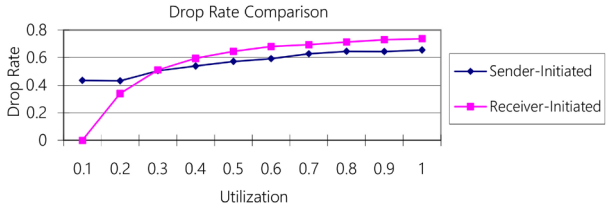

Figure 10. Drop rate comparison for sender-initiated protocol and receiver-initiated protocol.

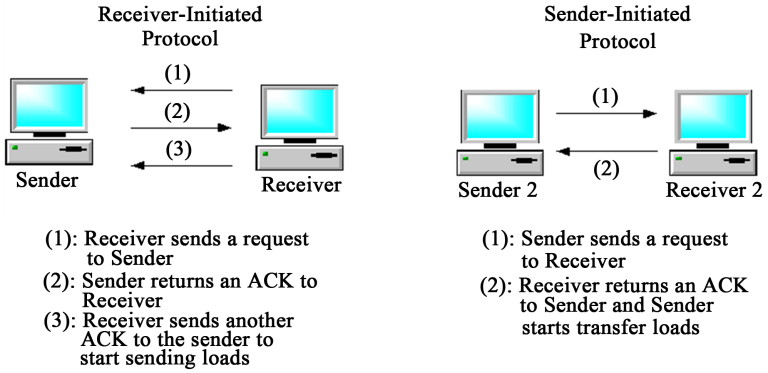

Figure 11. Steps and number of ACK for receiver-initiated and sender-initiated protocols.

6. Conclusions

This paper proposes a Fuzzy Logic Receiver-initiated load balancing protocol, which uses fuzzy logic control for load transfer. This protocol is based on a logical hierarchical structure that locates nodes for load transfer dynamically. To save the bandwidth of the network, multicast is used in the protocol. The simulation results showed that the performance of Receiver-initiated load balancing protocol is better than the BID methods for the data considered, and is better than the Non-Fuzzy Logic Receiver-initiated protocol when utilization is less than 0.7. The Fuzzy Logic Receiver-initiated protocol has the same system performance as the Non-Fuzzy Logic Receiver-initiated protocol when system utilization is higher than 0.7. All the results support that the system architecture and the protocol used in this study result in better system performance compared to some conventional load balancing methods.

Simulation results also showed that the Fuzzy Logic Receiver-initiated protocol is stable and smooth because no fixed threshold levels are used and there is a smooth transition from lightly to moderately, and to heavily loaded status and vice versa. The results also concluded that Fuzzy Logic Sender-initiated protocol has better system performance than the Fuzzy Logic Receiver-initiated protocol with the data and simulation model used.

Conflicts of Interest

The author declares no conflicts of interest regarding the publication of this paper.

References

[1] Sinha, P.K. (1996) Distributed Operating Systems: Concepts and Design. IEEE Press, New York. https://doi.org/10.1109/9780470544419

[2] Chulhye, P. and Kuhl, J.G. (1995) A Fuzzy-Based Distributed Load Balancing Algorithm for Large Distributed Systems. *Autonomous Decentralized Systems*, April 1995, 266-273.

[3] Huang, M., Hosseini, S.H. and Vairavan, K. (2002) Load Balancing in Computer Networks. *Proceedings of ISCA* 15*th International Conference on Parallel and Distributed Computing Systems*, *Special Session in Network Communication and Protocols*, Louisville, 19-21 September 2002, 331-336.

[4] Eager, D.L., Lazowska, E.D. and Zahorjan, J. (1986) Adaptive Load Sharing in Homogeneous Distributed Systems. *IEEE Transactions on Software Engineering*, **12**, 662-675. https://doi.org/10.1109/TSE.1986.6312961

[5] Hosseini, S.H. (2000) Special Issue on Load Balancing. *Cluster Computing Journal*, **3**, No. 2.

[6] Hosseini, S.H., Litow, B., Malkawi, M. and Vairavan, K. (1987) Distributed Algorithms for Load Balancing in Very Large Homogeneous Systems. *Proceedings of ACM-IEEE Fall Joint Computer Conference*, Vol. 29, 397-404.

[7] Hosseini, S.H., Litow, B., Malkawi, M. and Vairavan, K. (1990) Analysis of a Graph Coloring Distributed Load Balancing Algorithm. *Journal of Parallel and Distributed Computing*, **10**, 160-166. https://doi.org/10.1016/0743-7315(90)90025-K

[8] Kremien, O. and Kramer (1992) Methodical Analysis of Adaptive Load Sharing Algorithms. *IEEE Transactions on Parallel and Distributed Systems*, **3**, 747-760. https://doi.org/10.1109/71.180629

[9] Kumar, A. (1989) Adaptive Load Control of the Central Processor in a Distributed System with a Star Topology. *IEEE Transactions on Computers*, **38**, 1502-1512. https://doi.org/10.1109/12.42120

[10] Lin, F.C.H. and Keller, R.M. (1987) The Gradient Model Load Balancing Method. *IEEE Transactions on Software Engineering*, **13**, 32-38. https://doi.org/10.1109/TSE.1987.232563

[11] Mirchandaney, R., Towsley, D. and Stankovic, J.A. (1989) Adaptive Load Sharing in Heterogeneous Systems. In: *Distributed Computing Systems*, IEEE CS Press, Los Alamitos, 198-306.

[12] Mirchandaney, R., Towsley, D. and Stankovic, J.A. (1989) Analysis of the Effects of Delays on Load Sharing. *IEEE Transactions on Computers*, **38**, 1513-1525. https://doi.org/10.1109/12.42124

[13] Wolffe, G.S. (1998) Scheduling and Load Balancing for Distributed Systems. PhD Dissertation, University of Wisconsin, Madison.

[14] Nandagopal, M., Gokulnath, K. and Uthariaraj, V. (2010) Sender Initiated Decentralized Dynamic Load Balancing for Multi Cluster Computational Grid Environment. *Proceedings of the* 1*st Amrita ACM-W Celebration on Women in Computing in India*, September 2010, Article No. 63. https://doi.org/10.1145/1858378.1858441

[15] Ross, T.J. (1995) Fuzzy Logic with Engineering Applications. McGraw Hill, New York.

[16] UCB/LBNL/VINT Network Simulator. http://www.isi.edu/nsnam/ns/index.html

[17] Postel, J. (1980) User Datagram Protocol. RFC 768, USC/Information Sciences Institute, Marina del Rey. https://doi.org/10.17487/rfc0768

[18] Smith, R. (1980) The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver. *IEEE Transactions on Computers*, **29**, 1104-1113. https://doi.org/10.1109/TC.1980.1675516

[19] Eager, D.L., Lazowska, E.D. and Zahorjan, J. (1985) A Comparison of Receiver-Initiated and Sender-Initiated Adaptive Load Sharing. *ACM SIGMETRICS Performance Evaluation Review*, *Proceedings of the* 1985 *ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, August 1985, 1-3. https://doi.org/10.1145/317786.317802